

A database approach to information retrieval: The remarkable relationship between language models and region models

Djoerd Hiemstra and Vojkan Mihajlović

University of Twente

Centre for Telematics and Information Technology

P.O. Box 217, 7500 AE Enschede, The Netherlands

{d.hiemstra,v.mihajlovic}@utwente.nl

Abstract

In this report, we unify two quite distinct approaches to information retrieval: region models and language models. Region models were developed for structured document retrieval. They provide a well-defined behaviour as well as a simple query language that allows application developers to rapidly develop applications. Language models are particularly useful to reason about the ranking of search results, and for developing new ranking approaches. The unified model allows application developers to define complex language modeling approaches as logical queries on a textual database. We show a remarkable one-to-one relationship between region queries and the language models they represent for a wide variety of applications: simple ad-hoc search, cross-language retrieval, video retrieval, and web search.

1 Introduction

The introduction of the relational model by Codd in 1970 [13] marks one of the success stories of computer science. The relational model laid the path for the development of relational database systems: general software tools for management of data with a well-understood and well-defined behaviour. They allow application developers to rapidly develop application programs that are easy to understand, document and teach [16]. Indeed, saying “databases” is saying “relational”: Virtually any introductory book or course on databases will teach the basics of the relational data model and SQL.

It can be argued that information retrieval is still at the stage where databases were in the 1960’s. There is no such thing as an equivalent of the relational model for information retrieval systems. Introductory books and courses on information retrieval [5, 45] will teach the student several information retrieval models – mostly focusing on different ranking strategies – each with its own strengths and weaknesses. Developing a retrieval application or deploying a search engine requires applications to call non-standard application program interfaces (APIs) and use non-standard query languages.

As an example, the Terrier system, a research information retrieval system developed by the University of Glasgow [41], is based on the so-called divergence of randomness models [1]. Terrier provides APIs for indexing and querying. To use the Terrier indexing API on a non-standard collection (Terrier comes with some fully implemented APIs, for instance for HTML documents), the application developer needs to create an object which implements the

collection interface. This will find all the files it has to process, and opens each one to create a document object which identifies which tags (or other byte sequences) act as document delimiters. Applications programs that work with this setup will be logically impaired if the file locations or document format (for instance the XML DTD) need to be changed. Or, in analogy with Codd’s [13] analysis of the database systems from the 1960’s: The retrieval system does not provide *access path independence*.

As another example, the Lemur toolkit [40] is a research retrieval system that is specifically designed to support research in language modeling [24, 36, 43]. The toolkit supports a broad range of different applications of information retrieval such as ad hoc retrieval, distributed retrieval, cross-language retrieval, etc. Lemur supports at least four different index types, each supporting different kinds of queries. For instance, some indexes include word positions to allow proximity queries, whereas others only allow very basic functionality. Application programs that work with one kind of index might be logically impaired if the index type is changed. In analogy with Codd [13], the retrieval system does not provide *indexing independence*.¹

In the past, we have used systems like Terrier and Lemur to research new applications of information retrieval technology such as cross-language retrieval [23], web retrieval [28], and video shot retrieval [25]. To develop such retrieval approaches, it was necessary to reimplement parts of the existing system: reimplementing APIs, introducing new APIs, introducing new query languages, and even introducing new indexing and storage structures. In this report, we present a framework that supports all such approaches by means of a simple yet powerful query language (similar to SQL or relational algebra) that hides the implementation details of retrieval approaches from the application developer. As such, the system provides access path independence and indexing independence.

There have been other attempts to develop approaches to information retrieval that provide data independence. For instance, Schek [49] describes methods for integrating databases and information retrieval systems where application programs and queries are not aware of access paths and indexes. Fuhr [19] describes a layered system design for information retrieval systems following the ANSI/SPARC model [55], distinguishing a physical (internal) layer, a conceptual layer and an external layer. The system might process queries in several ways, such as directly by an index, or by using an index as a filter with an additional scan of the filtered results. Probabilistic relational algebra or probabilistic Datalog (see [18] for an overview) might serve as conceptual query languages in such systems. An example of a system that implements this approach is HySpirit [21]. In this report we introduce an alternative for probabilistic relational algebra and probabilistic Datalog that is much closer to existing models of information retrieval.

1.1 Region models

Motivated by the data independence issues described above, Burkowski [11] proposes a mathematical framework which he called the *containment model* that operates on sets of contiguous extents. We will call extents *regions* in this report, and the model *region model*. A region might be a word, a phrase, a text element such as a title, or a complete document. Burkowski’s model comes with a small number of basic operators on sets of regions, the most important ones being SN (select narrow) and SW (select wide). A search for chapters

¹Codd identified one more type of data independence: ordering independence. As textual data is inherently ordered we are not concerned with ordering independence.

containing the word “databases” would be expressed as `<chapter> SW databases`, and if the application program only needs to put the chapter’s title on the screen, the query would be `<chapter.title> SW (<chapter> SW databases)`. In Burkowski’s framework, the application program does not know how a text collection and its index facilities are managed. The complexity of the retrieval system is encapsulated in a module that only responds to simple command strings like the ones above. Similar frameworks are introduced by Salminen and Tompa [47], Clarke et al. [12], Baeza-Yates and Navarro [4], Consens and Milo [14], and Jaakkola and Kilpelainen [26]. We will call the models underlying these approaches *region models* in this report.

Unlike Codd’s relational model for databases, the region models above did not have a big impact on the information retrieval research community, nor on the development of new retrieval systems. The reason for this is quite obvious: region models do not explain in anyway how search results should be ranked. In fact, most region models are not concerned with ranking at all; one might say they – like the relational model – are actually data models instead of information retrieval models. Region model approaches that do address ranking, like Burkowski’s model [11] and the approach by Masuda et al. [32], only include it as an after-thought: Retrieve first, then rank with some standard retrieval model such as a vector space model using *tf.idf* weights [48].

1.2 Language models

If anything, an approach to information retrieval has to address the ranking of search results. Ranking is the single most important feature of a search engine, and information retrieval modeling almost exclusively focuses on ranking (see e.g. [5, Chapter 2]). Traditionally, developing ranking strategies involves engineering, fitting and tuning term weighting approaches to improve experimental results [48], although there are some notable exceptions, for instance the probabilistic model by Robertson and Sparck-Jones [46]. A more recent approach that does not require lots of fitting and tuning are statistical language models for information retrieval [24, 36, 43]. Language models assign a probability to a piece of text. They are built for each document: Each document model assigns a probability to a text query, and documents are ranked accordingly. Language models have been applied to a wide variety of retrieval problems, such as simple ad-hoc search [24, 27, 36], cross-language retrieval [7, 23, 30, 56], video retrieval using speech transcripts [15, 25], and web search [27, 28, 39]. Examples of these applications will be shown in Section 3.

1.3 Unifying region models and language models

In this report we introduce an approach to information retrieval that fully integrates region models and language models. The approach allows application developers to define complex language modeling approaches as logical region queries on a textual database. We show a remarkable one-to-one relation between region queries and the language models they represent for the four retrieval problems mentioned above: ad-hoc search, cross-language retrieval, video retrieval, and web search. The report is organised as follows. In Section 2 we introduce the combined region/language model. Section 3 illustrates the application of the model by relating probability measures to region queries. Finally in Section 4 we present future work and relate the approach to current work on XML query languages and XML database systems.

2 A region model for text databases and a query language

This section briefly introduces the unified region/language model. The definitions closely follow Burkowski’s model [11], which we extend with region scores similar to the score region algebra we used for XML information retrieval [31].

A **textual database** consists of a finite sequence of **words** w_1, w_2, \dots, w_{n-1} , where w_i is used to denote the word on position i in the database. Additionally, the textual database consists of a hierarchy of text **elements**. Both words and elements are identified by the word positions in the database. Text elements are sequences of words that have a particular significance in the database. For example, a database with recipes will have text elements “ingredients”, “quantities”, “instructions”, etc., typically marked up as XML.

A **scored region** r is defined by two integers $r.start$ and $r.end$ ($1 \leq r.start < r.end \leq n$), and a float $r.score$ ($r.score > 0$).² The integers $start$ and end represent respectively the position of the first word that belongs to the contiguous region, and the position directly following the last word that belongs to the region. A region might be a text element, but also any other contiguous sequence of words. Note that the region $(i, i + 1, s)$ includes one (and only one) word w_i with a score s .

Retrieval from the textual database is done with a simple query language consisting of words, elements and five basic **operators**: CONTAINING, CONTAINED_BY, SCALE, AND, and OR. The language defines an algebra on sets of scored regions. Unlike Burkowski’s model [11], there are no additional constraints on sets of regions. We will now one-by-one define the language primitives in a rather informal way. For convenience, Figure 1 contains a more formal definition of the operators using SQL.

A word A single word, for example the query **banana**, produces a set of regions R , where each region $r \in R$ defines a position of the word in the textual database; $r.start$ being the position on which the word occurs, $r.end = r.start + 1$, and $r.score = 1$.

An element A single element, for instance the query **<recipe>** produces a set of regions R , where each region $r \in R$ is tagged as “recipe”, $r.start$ being the position of the first word of the XML element, $r.end$ being the position following the last word of the XML element, and $r.score = 1$.

R_1 **CONTAINING** R_2 The operator CONTAINING takes two sets of regions R_1 and R_2 , and produces the subset of regions from R_1 that contain at least one region from R_2 . For instance, the query **<recipe> CONTAINING banana** produces all regions tagged as “recipe” that contain at least one occurrence of “banana”. Inspired by language models, each “recipe” region is scored by the number of occurrences of “banana” in the region, divided by the length of the region (measured as $r.end - r.start$). Occurrences of “banana” are weighted by their length and by their score (of course, in the example query both length = 1 as well as score = 1); see Figure 1.

R_1 **CONTAINED_BY** R_2 The operator CONTAINED_BY takes two sets of regions R_1 and R_2 , and produces the subset of regions from R_1 that are at least contained by one region from R_2 . For instance, the query **<ingredient> CONTAINED_BY <recipe>** produces all ingredients that belong at least to one recipe. If a region from the left-hand side of the expression is nested in more than one region from the right-hand side of the expression, then the

²We intentionally use a notation that is close to that of the relational data model; see also Figure 1.

scores of those regions are added. This will be used in the next section to express the linear combination of several language models; see Figure 1.

f **SCALE** R The operator SCALE takes a float f and a set of regions R and produces all regions from R where each region $r \in R$ is scored as $f \cdot r.score$. For instance, the query `0.2 SCALE banana` produces the set of regions with the positions of the word “banana” all with a region score of 0.2; see Figure 1.

R_1 **AND** R_2 The operator AND takes two sets of regions R_1 and R_2 , and produces only those regions that are both in R_1 and R_2 , i.e., the intersection of both sets when ignoring the region scores. Each region in the result is scored by multiplying its scores in R_1 and R_2 . For instance, the query `(<recipe> CONTAINING banana) AND (<recipe> CONTAINING apple)` produces all regions tagged as “recipe” that contain both the word “banana” and the word “apple”, scored by the product of the scores of the respective regions; see Figure 1.

R_1 **OR** R_2 The operator OR takes two sets of regions R_1 and R_2 , and produces those regions that either are in R_1 , or in R_2 , i.e., the union of both sets when ignoring the region scores. For instance, the query `(<recipe> CONTAINING sugar) OR (<recipe> CONTAINING sweet)` produces all regions tagged as “recipe” that contain either the word “sugar” or the word “sweet” (or both). Regions keep their score, unless both sets contain the region, in which case the region is scored by adding its scores in R_1 and R_2 ; see Figure 1.

```
-- R1 CONTAINING R2
SELECT R1.start, R1.end, R1.score * SUM((R2.score *
  (R2.end - R2.start)) / (R1.end - R1.start)) AS score
FROM R1, R2
WHERE R1.start <= R2.start AND R1.end >= R2.end
GROUP BY R1.start, R1.end, R1.score

-- R1 CONTAINED_BY R2
SELECT R1.start, R1.end, R1.score * SUM(R2.score) AS score
FROM R1, R2
WHERE R1.start >= R2.start AND R1.end <= R2.end
GROUP BY R1.start, R1.end, R1.score

-- f SCALE R
SELECT R.start, R.end, f * R.score AS score
FROM R

-- R1 AND R2
SELECT R1.start, R1.end, R1.score * R2.score AS score
FROM R1, R2
WHERE R1.start = R2.start AND R1.end = R2.end

-- R1 OR R2
SELECT R.start, R.end, SUM(R.score) AS score
FROM (SELECT * FROM R1 UNION ALL SELECT * FROM R2) AS R
GROUP BY R.start, R.end
```

Figure 1: Definition of operators in SQL.

Figure 1 contains a definition of the operators using SQL, as a pragmatic means to provide

a *formal* definition of the region algebra operators without the need to get into specific mathematical notations. So, we show SQL definitions here for convenience, as we assume most readers are familiar with SQL. The definitions do not suggest in any way that the system should be implemented on top a relational databases system. We implemented the system – without the use of SQL – on top of MonetDB [31], but it might as well be implemented using traditional inverted file indexes on the file system.³

A natural application of the region model, is to support structured queries in an XML information retrieval system. The following query is an example XML information retrieval query formulated in NEXI. NEXI [54] stands for narrowed extended XPath, a query language that restricts XPath [8] by only allowing descendent axis steps, and that extends XPath by a special about operator that ranks the selected nodes by their estimated relevance to the query. NEXI is used to evaluate XML retrieval systems in the Initiative for the Evaluation of XML retrieval (INEX) [20]. Suppose we want to retrieve sections about “databases” from articles that mention “book review” in either the article title (atl) or the keywords (kwd):

```
//article[about(./(atl|kwd), book review)]//sec[about(., databases)]
```

This can be formulated as follows as a region query:

```
(<sec> CONTAINING databases) CONTAINED_BY (<article> CONTAINING  
(((<atl> OR <kwd>) CONTAINING book) CONTAINING review))
```

This approach is followed with success in INEX by the TIJAH system [31, 35]. The expression defines a ranking of the selected nodes. Rewriting the NEXI query to the region expression is not trivial, but relatively easy: TIJAH has a NEXI to region query parser.

In the next section we show the relationship between language modeling ranking definitions and region queries, similar to the relationship between NEXI queries and the region queries.

3 Logical queries for complex retrieval tasks

3.1 The simplest unigram language model

As said in the introduction, language models form a general approach to define ranking formulas for retrieval applications. A language model is assigned to every document. The language model of the document defines the probability that the document ‘generates’ the query. Documents are ranked by this probability. The simplest language modeling approach to information retrieval would be defined by Equation 1.

$$P(T_1, T_2, \dots, T_l | D) = \prod_{i=1}^l P(T_i | D) \quad (1)$$

It defines the probability of a query of length l given a document D as the product of the probabilities of each term T_i ($1 \leq i \leq l$) given D . A language model that takes a simple product of terms, i.e., a model that assumes that the probability of one term given a document does not depend on other terms, is called a *unigram* language model. To make this work,

³For readers that do want to implement this on top of a relational DBMS, please note that ‘R1.end’ clashes with the SQL reserved word ‘END’ in practical systems.

we have to define the basic probability measure $P(T|D)$; typically, it would be defined as the number of occurrences of the term T in the document D , divided by the total number of terms in the document D . For a practical query, say, *retrieve all documents about “db” and “ir”*, we would instantiate Equation 1 as follows:

$$P(T_1=\text{db}, T_2=\text{ir}|D) = P(T_1=\text{db}|D) \cdot P(T_2=\text{ir}|D) \quad (2)$$

The right-hand side of the equation corresponds to the following region expression.

$$(\text{<doc> CONTAINING db}) \text{ AND } (\text{<doc> CONTAINING ir}) \quad (3)$$

This can be shown as follows: The region expression $(\text{<doc> CONTAINING db})$ produces all documents ranked according to $P(T = \text{db}|D)$, i.e., all regions tagged as <doc> , ranked by the number of occurrences of db in those regions. Similarly, $(\text{<doc> CONTAINING ir})$ produces all documents ranked according to $P(T = \text{ir}|D)$. Finally, the operator **AND** results in the regions tagged as <doc> that are in both operand sets. The score of the result regions is defined as the product of the scores of the same regions in the operands. Here, and in the remaining examples in this section, we assume that <doc> regions do not nest inside each other.

We claim that there is a trivial way to rewrite the right-hand side of Equation 2 to Equation 3 while preserving the outcome. This can be shown by simply replacing $P(x|y)$ by $(y \text{ CONTAINING } x)$, and the multiplication in Equation 2 by **AND**. Regions that are assigned zero probability by the probability measure of Equation 2 are not retrieved by the region expression of Equation 3. So, the region expression selects all y for which $P(x|y) > 0$. If the probability measure assigns zero probability to a region then this implies that the corresponding region expression will not retrieve it; and, if a region is not retrieved by a region expression then this implies that its corresponding probability function assigns zero probability to it.

3.2 Linear interpolation smoothing

The simple language model presented in the previous section assigns zero probability to a document unless it contains all query terms. So, if none of the documents contains all terms, the system does not retrieve anything. This behaviour will be appropriate for many practical applications. In fact, it is the default behaviour of web search engines like Google and Yahoo.

For other applications, it might be undesirable to have empty results. When searching collections that are significantly smaller than the web, it is likely that precise queries will not retrieve anything. In practice, language modeling approaches therefore use a technique called “smoothing”, i.e., some probability mass is assigned to terms that do not occur in the document. The standard language modeling approach uses a mixture of the document model $P(T_i|D)$ with a general collection model $P(T_i|C)$ [7, 24, 29, 36, 37, 50], called *linear interpolation smoothing*.

$$P(T_1, T_2, \dots, T_l|D) = \prod_{i=1}^l ((1-\lambda)P(T_i|C) + \lambda P(T_i|D)) \quad (4)$$

The document model $P(T_i|D)$ assigns zero probability to terms that do not occur in the document D , but the collection model $P(T_i|C)$ assigns some probability to any term that occurs somewhere in the collection. The collection model probabilities are defined similar to the document model probabilities as: The number of occurrences of the term T in the total

collection C , divided by the total number of terms in the collection C . The approach needs a parameter λ ($0 < \lambda < 1$) which is set empirically.

For our example query, we need some value for λ to instantiate Equation 4. Suppose we decide $\lambda = 0.8$, then we would rank documents according to:

$$\begin{aligned} P(T = \text{db}, T = \text{ir} | D) = & \\ (0.2 \cdot P(T_1 = \text{db} | C) + 0.8 \cdot P(T_1 = \text{db} | D)) & \\ \cdot & \\ (0.2 \cdot P(T_2 = \text{ir} | C) + 0.8 \cdot P(T_2 = \text{ir} | D)) & \end{aligned} \quad (5)$$

The equation corresponds to the following region expression, where the text element `<root>` corresponds to the collection root, i.e., the whole database.

$$\begin{aligned} & (\text{<doc> CONTAINED_BY} \\ & \quad ((0.2 \text{ SCALE } (\text{<root> CONTAINING db})) \text{ OR } (0.8 \text{ SCALE } (\text{<doc> CONTAINING db})))) \\ & \text{AND} \\ & (\text{<doc> CONTAINED_BY} \\ & \quad ((0.2 \text{ SCALE } (\text{<root> CONTAINING ir})) \text{ OR } (0.8 \text{ SCALE } (\text{<doc> CONTAINING ir})))) \end{aligned} \quad (6)$$

This can be shown as follows: The region expression `(<root> CONTAINING db)` results in a set with the single region `<root>` with a score equal to the number of occurrences of `db` in `<root>`, i.e., $P(T|C)$. The `SCALE` operator will multiply the region with 0.2; and the `OR` will union the region with all document regions (with scores $P(T|D)$ as in the previous section), multiplied with 0.8 by the `SCALE` operator. Note, that the `OR` operator will not actually add $0.2 \cdot P(T = \text{db} | C)$ to $0.8 \cdot P(T = \text{db} | D)$: This will be done by the `CONTAINED_BY` operator: every document region on the left-hand side of this operator matches (because every document region is contained by the collection root). Document regions that are in the set `0.8 SCALE (<doc> CONTAINING db)` will get as their final score: $0.2 \cdot P(T = \text{db} | C) + 0.8 \cdot P(T = \text{db} | D)$; the others will get: $0.2 \cdot P(T = \text{db} | C)$. The same line of reasoning can be done for the part with the term `ir`. Finally, the `AND` operator combines both parts of the query as in the previous section.

Again, we claim there is a trivial way to rewrite the right-hand side of Equation 5 to Equation 6. This can be shown by simply replacing $P(x|y)$ by `(y CONTAINING x)`, the multiplication operator `‘.’` by `AND` if both operands are regions, or by `SCALE` if the first operand is a number; the addition operator `‘+’` by `OR`, and by putting “`z CONTAINED_BY`” in front of the expression, where z defines the elements that need to be retrieved.

It might be argued that this very last step – “putting `CONTAINED_BY` in front” – is not a trivial step, and we did not use it in the previous section. However, we might as well use it in the previous section: It is easy to show that `(<doc> CONTAINING db) AND (<doc> CONTAINING ir)` produces the same regions, with the exact same scores as `(<doc> CONTAINED_BY (<doc> CONTAINING db)) AND (<doc> CONTAINED_BY (<doc> CONTAINING ir))`, because the elements on the left-hand side of both `CONTAINED_BY` operators all have unit score, and because elements on the left-hand side are nested in at most one region from the right-hand side of the `CONTAINED_BY` operator. So, the general procedure that rewrites probability measures to region expressions should use the `CONTAINED_BY` operator for every query term. Equivalences between region expressions will be addressed briefly in Section 4.1.

3.3 Video shot retrieval using speech transcripts

Now that we showed linear interpolation smoothing, it is easy to generalise this to any linear combination of language models. Such models have been quite successful in spoken document

retrieval for retrieving video shots [15, 25], where videos are modeled as sequences of scenes, each consisting of sequences of shots. The language model mixes four different levels of the video hierarchy: shots, scenes, complete videos and the total collection as:

$$P(T_1, T_2, \dots, T_l | \text{Shot}) = \prod_{i=1}^l (\alpha P(T_i | C) + \beta P(T_i | \text{Video}) + \gamma P(T_i | \text{Scene}) + \delta P(T_i | \text{Shot})) \quad (7)$$

where $\alpha + \beta + \gamma + \delta = 1$. The main idea behind this approach is that a good shot contains the query terms, and is part of a scene that contains the query terms, which is part of a video that contains even more of the query terms. Suppose we are looking for the exact shots in a collection of videos where a knight says “ni”,⁴ and we take $\alpha = 0.18$, $\beta = 0.02$, $\gamma = 0.4$, and $\delta = 0.4$ then the shots would be ranked according to:

$$\begin{aligned} P(T=\text{ni} | \text{Shot}) = & \\ & (0.18 \cdot P(T=\text{ni} | C) + 0.02 \cdot P(T=\text{ni} | \text{Video}) \\ & + 0.4 \cdot P(T=\text{ni} | \text{Scene}) + 0.4 \cdot P(T=\text{ni} | \text{Shot})) \end{aligned} \quad (8)$$

which corresponds to the following region expression.

$$\begin{aligned} & \langle \text{shot} \rangle \text{ CONTAINED_BY} \\ & ((0.18 \text{ SCALE } (\langle \text{root} \rangle \text{ CONTAINING ni})) \text{ OR } (0.02 \text{ SCALE } (\langle \text{video} \rangle \text{ CONTAINING ni})) \\ & \text{ OR } (0.4 \text{ SCALE } (\langle \text{scene} \rangle \text{ CONTAINING ni})) \text{ OR } (0.4 \text{ SCALE } (\langle \text{shot} \rangle \text{ CONTAINING ni}))) \end{aligned} \quad (9)$$

Showing that the region expression of Equation 9 retrieves and ranks video shots according to Equation 8 is done as in the previous section.

3.4 Web retrieval with page priors

For web retrieval, non-content information like the number of hyperlinks pointing to a web page, or the form of the URL are good indicators of the importance of a page. Such approaches can be modeled by so-called document priors $P(D)$ that do not depend on the query [27, 28, 39]. Document priors are calculated once for the entire collection, stored in the system and then used to enhance retrieval results for every query. A good example of such an approach is Google’s PageRank algorithm [10].

Document priors are motivated as follows. Instead of ranking documents by the probability that they generate the query, it makes more sense to rank them by $P(D | T_1, T_2, \dots, T_l)$: The probability that D is relevant given the query T_1, T_2, \dots, T_l of length l . According to Bayes’ rule:

$$\begin{aligned} P(D | T_1, T_2, \dots, T_l) &= \frac{P(D) \cdot P(T_1, T_2, \dots, T_l | D)}{P(T_1, T_2, \dots, T_l)} \\ &\propto P(D) \cdot \prod_{i=1}^l P(T_i | D) \end{aligned} \quad (10)$$

The denominator, $P(T_1, T_2, \dots, T_l)$, does not depend on D and can therefore be dropped, but document prior, $P(D)$, cannot be dropped unless it is uniformly distributed over all

⁴From the movie “Monty Python and the Holy Grail”

documents. Suppose we are looking for the entry page of Google. Documents will be ranked as follows.

$$P(D|T=\text{google}) \propto P(D) \cdot P(T=\text{google}|D) \quad (11)$$

To follow this approach, the system needs to have some means to store text elements with their prior probability. Suppose an application program calculated the PageRank of each crawled web page resulting in probabilities $P(D)$ (or any number proportional to the probabilities, see [10]) for each document region, which is stored as `$PageRank`. The dollar sign is used to denote a region set that is stored by the system for later use. The set is used in the query as follows.

$$\text{\$PageRank AND (<doc> CONTAINING google)} \quad (12)$$

We believe the correspondence between Equation 11 and 12 is obvious. As before, the query `\$PageRank AND (<doc> CONTAINED_BY (<doc> CONTAINING google))` would be a more general query that produces the exact same results.

3.5 Cross-language information retrieval

In cross-language information retrieval, a collection in one language, e.g. English, is searched by querying it in another language, e.g. Dutch. A language modeling approach to cross-language retrieval ranks documents by the probability $P(S_1, S_2, \dots, S_l|D)$ of generating a Dutch query S_1, S_2, \dots, S_l of length l from the English document D . This is modeled by the following procedure: first an English word T is generated from a document with probability $P(T|D)$, then the English term is translated to Dutch independently from the document it was generated from, so with probability $P(S|T)$, resulting in [7, 23, 56]:

$$P(S_1, S_2, \dots, S_l|D) = \prod_{i=1}^l \sum_{j=1}^V (P(S_i|T_j)P(T_j|D)) \quad (13)$$

where $P(T_j|D)$ is again the document language model, and $P(S_i|T_j)$ is a translation model defining the probabilities of the source language words (for instance Dutch in case of a Dutch query) given the target language words (English if the collection being searched is English), and where V is the size of the target language vocabulary. Such a model is used as follows: Given a Dutch query S_1, S_2, \dots, S_l , every word might have several possible translations in English. Suppose we want to use the Dutch query `gebroken hart` (English: “broken heart”) to search for English documents. The application program would consult its dictionary to determine that there are two possible English translations for the Dutch word “gebroken”: “broken” and “fractured”. The probability of translating “broken” to “gebroken”, i.e. $P(S = \text{gebroken}|T = \text{broken})$ might be estimated as 1.0, for instance because from example texts we know that the English word “broken” is always translated to “gebroken”; and the probability of translating “fractured” to “gebroken”, i.e. $P(S = \text{gebroken}|T = \text{fractured})$ might be estimated as 0.2 (note that the two probabilities do not need to sum up to 1). In this case, an instantiation of Equation 13 would be:

$$\begin{aligned} P(S_1=\text{gebroken}, S_2=\text{hart}|D) = & \\ & (1.0 \cdot P(T_1=\text{broken}|D) + 0.2 \cdot P(T_1=\text{fractured}|D)) \\ & \cdot \\ & (0.5 \cdot P(T_2=\text{heart}|D) + 0.1 \cdot P(T_2=\text{ticker}|D)) \end{aligned} \quad (14)$$

So, the sum over the whole target language vocabulary will in practice be a sum over the possible translations only (those for which $P(S|T) > 0$). The probability function corresponds to the following region expression.

$$\begin{aligned}
& ((1.0 \text{ SCALE } (<\text{doc}> \text{CONTAINING broken})) \text{ OR } (0.2 \text{ SCALE } (<\text{doc}> \text{CONTAINING fractured}))) \\
& \text{AND} \\
& ((0.5 \text{ SCALE } (<\text{doc}> \text{CONTAINING heart})) \text{ OR } (0.1 \text{ SCALE } (<\text{doc}> \text{CONTAINING ticker})))
\end{aligned} \tag{15}$$

Equation 15 can be generated from 14 as shown in the previous sections.

4 Discussion, open issues and future work

In this report, we presented a unified region model / language model approach and showed its expressiveness for a wide range of applications of language modeling: ad-hoc retrieval, smoothing, video retrieval, web search and cross-language retrieval. In the past, we have developed separate prototype retrieval systems for these approaches. Developing these prototype systems meant we had to reimplement parts of our system: reimplementing APIs, introducing new APIs, introducing new query languages, introducing new indexes, introducing new storage structures, etc. This report shows that such approaches can be supported by a single retrieval system that responds to a simple query language that hides implementation details of information retrieval approaches from the application developer.

The relationship between the region queries and the language modeling probability functions might seem trivial because we “hard-wired” the language modeling probability definition in the CONTAINING operator, but we believe it is remarkable: Note that the language modeling probability functions are arithmetic expressions that define the probability of a single document D . However, the region queries are algebraic expressions for processing *sets* of documents (regions) instead of single documents. Since the region query language forms a “bulk algebra”, experiences from relational database system design can be used to develop efficient implementations of such a system, possibly up to a point where applications run as fast as, or possibly even faster than, the dedicated prototypes we developed in the past.

4.1 Query optimization

The queries presented in Section 2 are close to the language modeling probability functions. However, there exist alternative expressions of the queries that produce equivalent results but that might be easier to process by the system. Based on a study into equivalence relations for region models [34], we conjecture that the following expressions are alternatives for the expressions presented in Section 2: $(<\text{doc}> \text{CONTAINING db}) \text{ CONTAINING ir}$ is an alternative for Equation 3; $(<\text{doc}> \text{CONTAINED_BY } (((0.2 \text{ SCALE } <\text{root}>) \text{ OR } (0.8 \text{ SCALE } <\text{doc}>))) \text{ CONTAINING db}) \text{ CONTAINED_BY } (((0.2 \text{ SCALE } <\text{root}>) \text{ OR } (0.8 \text{ SCALE } <\text{doc}>))) \text{ CONTAINING ir}$ is an alternative for Equation 6; $<\text{shot}> \text{ CONTAINED_BY } (((0.18 \text{ SCALE } <\text{root}>) \text{ OR } (0.02 \text{ SCALE } <\text{video}>) \text{ OR } (0.4 \text{ SCALE } <\text{scene}>) \text{ OR } (0.4 \text{ SCALE } <\text{shot}>)) \text{ CONTAINING ni}$ is an alternative for Equation 9; $\$ \text{PageRank CONTAINING google}$ is an alternative for Equation 12; finally $(<\text{doc}> \text{CONTAINING } (\text{broken OR } (0.2 \text{ SCALE } \text{fractured}))) \text{ CONTAINING } ((0.5 \text{ SCALE } \text{heart}) \text{ OR } (0.1 \text{ SCALE } \text{ticker}))$ is an alternative for Equation 15.

Additionally, query optimization would involve choosing concrete evaluation methods attached to each operation, estimating the costs of each method, and choosing the fastest plan. Ramírez and De Vries [44] present preliminary results.

4.2 Towards existing XML query languages

It can be argued that region models are simple predecessors of models underlying XML query languages like XPath [8] and XQuery [9]. That is, operators like `CONTAINED_BY` and `CONTAINING` can be seen as ancestor and descendent axis steps, as well as the function `fn:contains` in XPath. It would be relatively easy to add other XPath axis steps to the query language if we specify how regions are nested, for instance by requiring that a region has a *level* (the depth in the XML tree) as well as a start, end, and score.

XML and its subsequent standards like XPath and XQuery have initiated a lot of research into XML database systems with dedicated workshops and symposia like DataX [33] and XSym [6]. Our implementation of the region approach is quite similar to implementations of XML databases that use relational database technology and a numbering of the XML nodes [52]. Interestingly, the word positions that belong to the region start and region end of an XML element are respectively in pre-order and post-order as in the XML database implementation proposed by Grust [22]. Our prototype system TIJAH uses part of the code of the PathFinder XML database system [53]. In the future, both systems might be integrated following the XQuery full-text standard [2, 3].

4.3 Towards new applications of XML

Some people have argued that existing XML query languages like XPath [8] and XQuery [9] are too powerful for simple XML information retrieval functionality [54]. Others have argued that existing query languages are not powerful enough. For instance Ogilvie [38] illustrates a system that answers queries like “Who killed Abraham Lincoln” by a query that returns those `<person>` elements that directly precede the word `killed`, which directly precedes another `<person>` element containing `lincoln`. Such a query would be hard, if not impossible, to express in existing XML query languages. A solution might be the introduction of a special gluing operator in our region model approach, let’s call it ADJ for “adjacent”, which can glue regions to form bigger regions. Such an operator might be used for phrases, but also to glue for instance two paragraphs together to form a region that spans two paragraphs. We have implemented such a gluing operator in our video retrieval system that, lacking a reliable scene detector, glues adjacent shots together to represent a scene [25].

4.4 Beyond XML

Ogilvie [38] also makes a case for allowing several hierarchies of possibly overlapping elements which combined would no longer form a tree. This need is illustrated as well by Burkowski [11], by people studying the bible [17], and it is picked up by several initiatives to extend XML [42, 51]. The region approach described here would support querying of such representations quite naturally.

Acknowledgements

Djoerd Hiemstra was supported by the Dutch BSIK program MultimediaN: Semantic Multimedia Access. Vojkan Mihajlović was supported by the Netherlands Organisation for Scientific Research (NWO project 612.061.210). We like to thank Henk Ernst Blok for fruitful discussions on region algebras, and Maarten Fokkinga and Thijs Westerveld (CWI, Amsterdam) for helpful comments on the report.

References

- [1] G. Amati and C.J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems*, 20(4):357–389, 2002.
- [2] S. Amer-Yahia, C. Botev, S. Buxton, P. Case, J. Doerre, D. McBeath, M. Rys, and J. Shanmugasundaram. XQuery 1.0 and XPath 2.0 full-text working draft. Technical Report, World Wide Web Consortium, April 2005. <http://www.w3.org/TR/xquery-full-text/>.
- [3] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TexQuery: A full-text search extension to XQuery. In *Proceedings of the 13th conference on World Wide Web*, pages 583–594, 2004.
- [4] R. Baeza-Yates and G. Navarro. Proximal nodes: A model to query document databases by content and structure. *ACM Transactions on Information Systems*, 15(4):401–435, 1997.
- [5] R.A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
- [6] Z. Bellahsene, T. Milo, M. Rys, D. Suciu, and R. Unland, editors. *Proceedings of the 2nd International XML Database Symposium (XSym), Lecture Notes in Computer Science 3186*. Springer, 2004.
- [7] A. Berger and J. Lafferty. Information retrieval as statistical translation. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 222–229, 1999.
- [8] A. Berglund, S. Boag, D. Chamberlin, M. Fernandez, M. Kay, J. Robie, and J. Simeon. XML path language (XPath) 2.0. Technical Report, World Wide Web Consortium, 2005. <http://www.w3.org/TR/xpath20/>.
- [9] S. Boag, D. Chamberlin, M. Fernandez, D. Florescu, J. Robie, and J. Simeon. XQuery 1.0: An XML query language. Technical Report, World Wide Web Consortium, 2005. <http://www.w3.org/TR/xquery/>.
- [10] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.
- [11] F.J. Burkowski. Retrieval activities in a database consisting of heterogeneous collections of structured texts. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 112–125, 1992.
- [12] C.L.A. Clarke, G.V. Cormack, and F.J. Burkowski. Algebra for structured text search and a framework for its implementation. *The Computer Journal*, 38(1):43–56, 1995.

- [13] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 1970.
- [14] M. Consens and T. Milo. Algebras for querying text regions. In *Proceedings of the ACM Conference on Principles of Database Systems (PODS)*, pages 11–22, 1995.
- [15] E. Cooke, P. Ferguson, G. Gaughan, C. Gurrin, G.J.F. Jones, H. Le Borgue, H. Lee, S. Marlow, K. McDonald, M. McHugh, N. Murphy, N.E. O’Connor, N. O’Hare, S. Rothwell, A.F. Smeaton, and P. Wilkins. TRECVID 2004 experiments in Dublin City University. In *Proceedings of the TRECVID workshop*, 2005.
- [16] C.J. Date. *An introduction to database systems*. Addison-Wesley, 1981.
- [17] S. DeRose. Markup overlap: A review and a horse. In *Proceedings of the fifth Conference on Extreme Markup Languages*, 2004.
- [18] N. Fuhr. Models for integrated information retrieval and database systems. *Data Engineering Bulletin. Special issue of integrating text retrieval and databases*, pages 3–13, 1996.
- [19] N. Fuhr. Towards data abstraction in networked information retrieval systems. *Information Processing and Management*, 35(2):101–119, 1999.
- [20] N. Fuhr, S. Malik, and M. Lalmas. Overview of the initiative for the evaluation of XML retrieval. In *Proceedings of the 3rd Initiative on the Evaluation of XML Retrieval (INEX)*. Springer Lecture Notes in Computer Science (LNCS 3493), 2005.
- [21] N. Fuhr and T. Rölleke. HySpirit - a probabilistic inference engine for hypermedia retrieval in large databases. In *Proceedings of the 6th International Conference on Extending Database Technology (EDBT)*, pages 24–38, 1998.
- [22] T. Grust. Accelerating XPath location steps. In *Proceedings of the ACM international conference on management of data (SIGMOD)*, pages 109–120, 2002.
- [23] D. Hiemstra and F.M.G. de Jong. Disambiguation strategies for cross-language information retrieval. In *Proceedings of the third European Conference on Research and Advanced Technology for Digital Libraries (ECDL)*, pages 274–293, 1999.
- [24] D. Hiemstra and W. Kraaij. Twenty-One at TREC-7: Ad-hoc and cross-language track. In *Proceedings of the seventh Text Retrieval Conference (TREC)*, pages 227–238. 1999.
- [25] T. Ianeva, L. Boldareva, T. Westerveld, R. Cornacchia, D. Hiemstra, and A.P. de Vries. Probabilistic approaches to video retrieval. In *Proceedings of the TRECVID workshop*, 2005.
- [26] J. Jaakkola and P. Kilpelainen. Nested text-region algebra. Technical Report CR-1999-2, Department of Computer Science, University of Helsinki, 1999.
- [27] J. Kamps, G. Mishne, and M. de Rijke. Language models for searching in web corpora. In *Proceedings of the 13th Text Retrieval Conference (TREC-13)*, 2005.
- [28] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, 2002.
- [29] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 111–119, 2001.

- [30] V. Lavrenko, M. Choquette, and W.B. Croft. Cross-lingual relevance models. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 175–182, 2002.
- [31] J. List, V. Mihajlović, G. Ramírez, A.P. de Vries, D. Hiemstra and H.E Blok. TIJAH: Embracing information retrieval methods in XML databases. *Information Retrieval Journal* 8(4):547–570, Kluwer, 2005.
- [32] K. Masuda. A ranking model of proximal and structural text retrieval based on region algebra. In *Proceedings of the ACL-2003 Student Research Workshop*, pages 50–57, 2003.
- [33] M. Mesiti, B. Catania, G. Guerrini, and A. Chaudhri. Report on the EDBT’04 workshop on database technologies for handling XML information on the web. *SIGMOD Record*, 33(2), 2004.
- [34] V. Mihajlović, D. Hiemstra, H.E. Blok, and P.M.G. Apers. An XML-IR-DB sandwich: Is it better with an algebra in between? In *Proceedings of the joint SIGIR workshop on XML, Information Retrieval and Databases*, pages 39–46, 2004.
- [35] V. Mihajlović, G. Ramírez, A.P. de Vries, D. Hiemstra and H.E. Blok. TIJAH at INEX 2004: Modeling phrases and relevance feedback. In *Proceedings of the Initiative on the Evaluation of XML Retrieval (INEX)*. Springer LNCS 3493, 2005.
- [36] D.R.H. Miller, T. Leek, and R.M. Schwartz. A hidden Markov model information retrieval system. In *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 214–221, 1999.
- [37] K. Ng. A maximum likelihood ratio information retrieval model. In *Proceedings of the eighth Text Retrieval Conference (TREC)*. 2000.
- [38] P. Ogilvie. Retrieval using structure for question answering. In *Proceedings of the Twente Data Management Workshop (TDM)*, 2004.
- [39] P. Ogilvie and J. Callan. Combining structural information and the use of priors in mixed named-page and homepage finding. In *Proceedings of the 12th Text Retrieval Conference (TREC)*, pages 177–184, 2004.
- [40] P. Ogilvie and J.P. Callan. Experiments using the Lemur toolkit. In *Proceedings of the tenth Text Retrieval Conference, TREC-10*, pages 103–108. 2002.
- [41] I. Ounis, G. Amati, V. Plachouras, B. He, C. MacDonald, and D. Johnson. Terrier information retrieval platform. In *Proceedings of the 27th European Conference on Information Retrieval, ECIR-05*, 2005.
- [42] W. Piez. Half-steps toward LMNL. In *Proceedings of the Conference on Extreme Markup Languages*, 2004.
- [43] J.M. Ponte and W.B. Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 275–281, 1998.
- [44] G. Ramírez and A.P. de Vries. Combining indexing schemes to accelerate querying XML on content and structure. In *Proceedings of the Twente Data Management Workshop (TDM)*, 2004.
- [45] C.J. van Rijsbergen. *Information Retrieval, second edition*. Butterworths, 1979.
- [46] S.E. Robertson and K. Sparck-Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

- [47] A. Salminen and F.W. Tompa. Pat expressions: An algebra for text search. In *Proceedings of the 2nd International Conference in Computational Lexicography, COMPLEX'92*, pages 309–332, 1992.
- [48] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.
- [49] H.J. Schek. Methods for the administration of textual data in database systems. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 218–235, 1980.
- [50] F. Song and W.B. Croft. A general language model for information retrieval. In *Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM)*, pages 316–321, 1999.
- [51] C.M. Sperberg-McQueen and C. Huitfeldt. Goddag: A data structure for overlapping hierarchies. In *Joint Conference of the ALLC and ACH*, 1999.
- [52] I. Tatrinov, S. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang. Storing and querying ordered XML using a relational database system. In *Proceedings of the ACM SIGMOD international conference on Management of data*, pages 204–215, 2002.
- [53] J. Teubner, P. Boncz, T. Grust, M. van Keulen, S. Manegold, and J. Rittinger. PathFinder: XQuery – the relational way. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2005.
- [54] A. Trotman and R.A. O’Keefe. The simplest query language that could possibly work. In *Proceedings of the 2nd Workshop of the INitiative for the Evaluation of XML retrieval (INEX)*, 2004.
- [55] D. Tsichritzis and A. Klug. The ANSI/X3/SPARC DBMS framework report of the study group on database management systems. *Information Systems*, 3:173–191, 1978.
- [56] J. Xu, R. Weischedel, and C. Nguyen. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*, pages 105–110, 2001.